Using movement primitives in interpreting and decomposing complex trajectories in learning-by-doing

Andrea Soltoggio, Andre Lemme and Jochen J. Steil[1]

Preprint accepted for publication in the
2012 IEEE International Conference on Robotics and Biomimetics (ROBIO 2012)

***Abstract*— Learning and reproducing complex movements is an important skill for robots. However, while humans can learn and generalise new complex trajectories, robots are often programmed to execute point-by-point precise but fixed patterns. This study proposes a method for decomposing new complex trajectories into a set of known robot-based primitives. Instead of reproducing accurately an observed trajectory, the robot interprets it as a composition of its own previously acquired primitive movements. The method attempts initially a rough approximation with the idea of capturing the most essential features of the movement. Observing the discrepancy between the demonstrated and reproduced trajectories, the process then proceeds with incremental decompositions. The method is tested on both geometric and human generated trajectories. The shift from a data-centred view to an agent-centred view in learning trajectories results in generalisation properties like the abstraction to primitives and noise suppression. This study suggests a novel approach to learning complex robot motor patterns that builds upon existing motor skills. Applications include drawing, writing, movement generation and object manipulation in a variety of tasks.**

I. INTRODUCTION

Humans and animals are capable of learning, perfecting and reproducing complex trajectories that allow them to perform a variety of tasks, from coordinated body movements to catching, and particularly in humans, object manipulation, writing and drawing. The mechanisms underlying motor skills, from the learning of basic primitives to their organisation in higher-level cognitive structures, are fundamental in understanding how humans accomplish advanced motor skills [1]. Object manipulation, skilful movements and generalised trajectories are considered fundamental in the evolution of intelligence and modern technology [2]. The autonomous learning of robotic movements and their organisation are an increasingly important research focus. Object manipulation and precise movements are implemented in industrial robots for manufacturing and production processes. However, a considerable limitation in such movements is that trajectories are often pre-programmed and executed point-by-point, therefore lacking a general and symbolic representation of the movement, as well as the capability of adapting and improving.

[1]are with the Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld Universitt, 33615 Bielefeld, Germany. asoltogg, alemme, jsteil at cor-lab.uni-bielefeld.de

Methods for generating robotic trajectories with accurate control of position, velocity and acceleration have been proposed. Among those, the minimum-jerk model [3], dynamic movement primitives (DMP) [4], [5], Gaussian mixture models (GMM) [6] and recurrent neural networks (RNN) [7], [8] have gained considerable attention. Reinforcement learning has also been applied for learning trajectories from demonstration [9], in which a reinforcement signal is used to decrease an error between demonstration and actual trajectory [10], [11]. So far most studies have focused on the generation of accurate motor primitives that represent a repertoire of basic movements. However, one intrinsic feature and limitation of motor primitives is that they generate basic rather than arbitrary long and convoluted trajectories. As a consequence, recent research has focused on the problem of combining simple primitives to form more complex and longer trajectories [12], [13], [14], [10], [15]. Some approaches, e.g. [12], [13], [16], focus on the demonstrated trajectory that is analysed and decomposed, employing polynomial decomposition [12], Hidden Markov Models [16], Non-Negative Matrix Factorisation [13] and detection of critical points [17]. Other methods instead, particularly those based on reinforcement learning, assume the presence of an agent and a process of learning-by-doing [14], [10], [15]. Finally, methods have been proposed to create natural-looking shapes by joining movement sequences [18], [19] and co-articulation [20]. Algorithms that combine primitive or shape-identification, trajectory segmentation and on-line learning have also be proposed [21], [17], [15] to integrate various subproblems in more capable learning algorithms.

As opposed to previous studies, the proposed method focuses entirely on the segmentation process, assuming that primitives are previously learnt by the agent. This separation between formation of primitives and decomposition of complex trajectories implies that the existing and well-established methods for movement generations cited above (e.g. DMP, RNN) can be employed in combination with the current algorithm. Therefore, the set of primitives can be freely chosen and used in combination with the proposed decomposition method. A second implication is that the set of basic primitives are the building blocks by means of which the learner interprets demonstrations. The features of the demonstrated trajectory, e.g. inflection points, points of discontinuous derivative, critical points, etc., that are subject

of extensive analysis in the other decomposition algorithms (e.g. [16], [13], [17]) are not the focus in the present method. These features of the demonstration emerge autonomously in the reproduced trajectory as the imitation learning takes place.

One fundamental and novel aspect of the proposed method is that the decomposition of complex trajectories initiates as a rough approximation based on one single movement primitive. Interestingly, a complex trajectory encompassing many convoluted parts, e.g. a handwritten long word, is unlikely to be adequately represented by one single stroke. Yet this counter-intuitive position proves to be a fundamental step in the iterative process. The points of decomposition are progressively discovered during learning. At each iteration, the part of the reproduction with the maximum discrepancy with the demonstrated trajectory is considered for improvement. The accuracy of the learnt trajectory depends on the set of primitives, and not on the precision or noise in the demonstrated data. Therefore, this agent-centred view [22] has the advantage of reducing over-fitting by finding a symbolic (i.e. primitive-based) representation of a trajectory. The incremental learning from preexisting primitives to their combination in more complex trajectories makes also the algorithm particularly plausible from a biological perspective. In fact, advanced motor skills in humans have been shown to be represented as a combination of more basic skills [1].

The proposed decomposition algorithm lends itself to promising extensions including learning trajectories from multiple examples, hand-writing recognition, decomposition of complex movement patterns for manipulation and combination of skills. The decomposition algorithm is explained in detail in the next section. The performance are illustrated with tests on simple to complex trajectories to assess the accuracy and reliability (Sec. III). The implications and possible extensions are then discussed in Sec. IV before the conclusion in Sec V.

II. Iterative decomposition

Regardless of the length and complexity of a given trajectory, the counter-intuitive and original position in this study is that only the start and end points are initially considered as extremities of one single primitive. The agent searches among its own primitives the one that minimises the maximum distance between the demonstrated trajectory and the primitive itself. This is effectively a first solution. The point of maximum error is identified as a candidate decomposition point. This simple heuristic is not based on an optimality measure, which is difficult to infer in an iterative process. The idea is rather that of identifying prominent discrepancies between the current reproduction and the demonstrated trajectory. Once the trajectory is decomposed into two parts, the process is reiterated on both subparts. When a candidate decomposition does not bring an improvement to the maximum error on either sides, the process stops.

Fig. 1 illustrates a three-step decomposition of a trajectory.

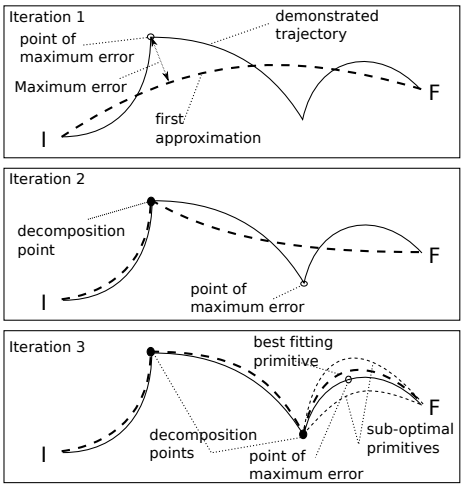

Fig. 1. Graphical illustration of the iterative decomposition process. Iteration 1 begins with one single primitive selected from a set of pre-learnt primitives, as approximation of the global trajectory between the initial (I) and end (E) points. Such approach is counter intuitive because one single primitive is not likely to result in a good approximation, especially for long and convoluted demonstrated trajectories. However, such initial step helps in identifying a point of maximum error and iterate the procedure to perform a decomposition. In Iteration 2, two better fitting primitives are identified, as well as a new point of maximum error. In iteration 3, the demonstration is well approximated by the sequence of primitives. The plot shows the best fitting primitives and two sub-optimal primitives for the third segment.

A. Using sets of primitives

The proposed method works in combination with a set of prelearnt primitives, which can be generated by means of a variety of algorithms independently of the present decomposition method, e.g. DMP or RNN. This feature is particularly important because it makes the proposed algorithm general and suited to build complex movements in combination with well established methods for primitive generations. Interestingly, the basic skills of the agent, expressed as the range or diversity of the basic motor primitives, determines the accuracy of approximation of arbitrary trajectories, but does not significantly affect the capability of the proposed method to perform decomposition. To show this concept, the current study considers two sets of primitives generated with the minimum-jerk model [3]: one basic set with only seven symmetric primitives (Fig. 2A), and a more complex set with 51 primitives with symmetric and asymmetric shapes (Fig. 2B). Details are provided in the Appendix V-A. The current study analyses for brevity the performance of the decomposition method with these two sets of primitives, but further sets can be included and tested with the Matlab code provided as support material.

B. Abstracting from sequences of points to primitives

When decomposed finely, any trajectory can be represented as a sequence of close points united by straight lines. Such a decomposition, however, merely copies a demonstrated trajectory without generalising the overall shape of a movement. The method in this study instead attempts to mimic a generic trajectory with well defined primitives, which guarantee generality and can be devised to guarantee

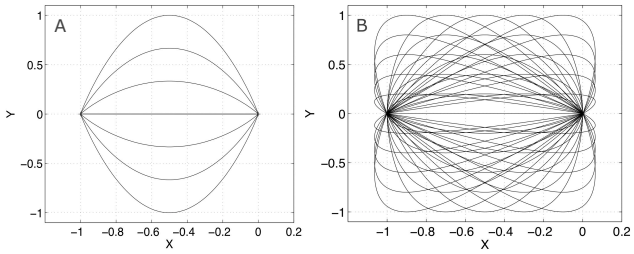

Fig. 2. Sets of primitives generated with the minimum jerk model [3]. (A) This small set of primitives contains one straight line and six curved lines. (B) This larger set of primitives contains 51 primitives both with symmetric and asymmetric geometry. Primitives are rotated and scaled to be fitted to the starting and ending point of a demonstration.

also efficiency and optimality. For example, the minimum-jerk model used in the current experiment guarantees energy minimisation and is biologically plausible [3].

How does the present algorithm identifies large primitives instead of infinitely increasing the number of decomposition points? Or alternatively, given the iterative nature of the process, when is the reproduction good enough to stop further decomposition? The problem is effectively both a classification problem (finding the best matching primitive) and an optimisation problem in which a balance between generality and number of decompositions is sought. In fact, a residual error must be accounted for to achieve a general and abstract representation. One possibility is to set a target low error, which stops the algorithm when such a low error is reached. However, an error parameter is not always a good measure to indicate how well an imperfect demonstration is abstracted into a primitive.

The alternative idea in the present algorithm is to associate with the set of primitives a *precision* parameter, which accounts for a second dimension in the primitive execution, as for example the thickness of a brush in a drawing task. The case is illustrated in Fig. 3, in which the primitive is shown with an associated precision (thick dashed line). In the experiments of this paper, the smaller set of 7 primitives has a precision value of 5% of the shortest side of the drawing area, while the more accurate 51-primitive set has a precision of 2.5%, i.e. the second dimension of a primitive is 2.5% of the drawing area. The precision parameter, or second dimension of a primitive, mimics real-world situations in which both demonstration and reproduction are perceived and executed with a limited precision. The algorithm can now perform a feature detection by observing whether the primitive intersects the demonstration or not. As illustrated in Figs. 3B and C, an intersection suggests the presence of a relevant feature that can be captured with further decomposition. If the best matching primitive does not intersect the demonstration, the current solution is considered the best generalisation and the algorithm stops. Therefore, the precision parameter is not an error threshold, but a value associated with the primitive set. This value can be interpreted as the thickness or precision of primitives.

Occasionally, the start and end points of a trajectory are

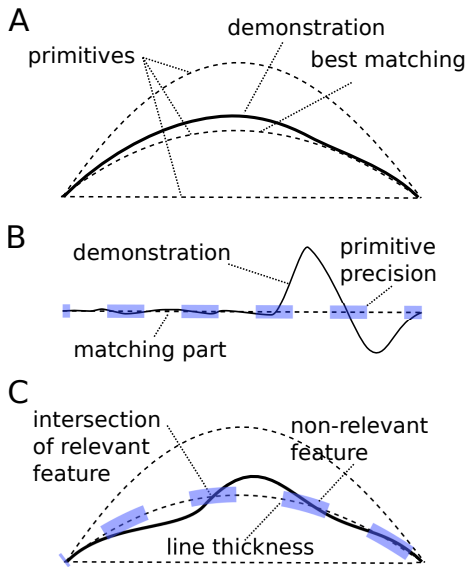

Fig. 3. Identification of shapes. (A) Three primitives are shown in the attempt to match a demonstrated trajectory. The best matching primitive is not a perfect reproduction, but rather a general abstraction of the demonstration. (B) A demonstration is matched to a straight primitive in which the second dimension, or precision, is shown. The left-most part of the trajectory, although not perfectly straight, does not intersect the two-dimension primitive, suggesting that no further features are present in the demonstration. The right-most part of the trajectory instead clearly intersects the primitive, indicating the utility of further decompositions. (C) Case as in A in which however the primitive intersects the demonstration, informing the algorithm that further decompositions are necessary.

relatively close, e.g. in the drawing of a circle. In these cases, because the algorithm always focuses on the start and end point, it is easy to detect that the trajectory leaves the areas of focus, i.e. the areas of all possible primitives. When this happens, further decompositions are necessary and the algorithm proceeds even if there are no intersections between the demonstration and the reproduction.

The nature of the proposed iterative method implies that the interpretation of the demonstration (i.e. the solution) varies and improves at each further decomposition. When a new decomposition point is inserted, effectively it questions the utility of the neighbouring points previously inserted. Therefore, at the insertion of a new decomposition point, primitives are also searched to skip the preceding and following points along the trajectory. Neighbouring decomposition points are eliminated if they do not reduce the maximum error or if the no-intersection criterion applies to the longer segments. This check adheres to the principle of generality by ensuring that each new point increases the number of primitives only when those effectively capture new features in the demonstration.

The criteria presented in this section encode the *intelligence* in the proposed decomposition algorithm, which intends to identify general shapes, rather than applying an exact and particular reproduction of a sequence of points. In the generalisation, primitives are used to represent finely sampled point-by-point demonstrations. The identification of primitives implies inevitably the classification of imprecise and noise-affected demonstration into well defined and exact

lines. Therefore, such a process causes the loss of precision from the demonstrated data. However, such a precision may not be descriptive of features of the demonstration. The heuristic presented in this section does not seek an optimality criterion nor an efficient error reduction. On the contrary, it proposes a method to abstract demonstration into concept primitives that generally describe a trajectory and best fit the skills of the agent or robot. The algorithm is presented as pseudo code in the Appendix.

III. Simulation Results

The current section reports the simulation results and performance of the algorithm applied to a variety of demonstrated trajectories, from simple to complex. The algorithm is implemented with Matlab® code that is provided as support material for this paper at the authors' website http://andrea.soltoggio.net/decomp.

A. Basic Examples

The decomposition algorithm is applied here on human and machine generated trajectories. These basic examples have the purpose of showing how the algorithm decomposes short trajectories. The examples in Fig. 4 show that the algorithm favours decompositions with few primitives, resulting in some cases in a residual error between demonstrated and reproduced trajectories. This error appears to derive from the more general trajectories chosen by the agent with respect to the irregular human generated data. In fact, the fourth row in Fig. 4 shows a machine generated trajectory which is accurately approximated. The decomposition with the set of 51 primitives (right column in Fig. 4) appears more accurate. The small set of 7 primitives instead captures the main features of the demonstrated trajectories, effectively achieving a higher level of abstraction. The implication is that in front of complex and precise trajectories, agents or robots with few primitives can nevertheless utilise the algorithm to decompose a complex trajectory according to their basic motor skills.

It is important to note that the primitives are executed sequentially without additional procedure to join them. Therefore, points of discontinuous derivative are noticeable where primitives join. However, while the current algorithm performs a decomposition, the reproduction of a complex trajectory can be integrated with combining algorithms to create smooth and natural-looking trajectories as proposed in [20], [19].

B. Performance on model-generated random sequences

An extensive test was executed to assess the performance of the algorithm on different trajectories. One hundred trajectories were constructed with sequences of three primitives from the 51-primitive set, applying random primitive lengths and rotations.

The decomposition with the 7-primitive set produced schematic and abstract interpretations as that in Fig. 5B. Interestingly, a straight primitive was frequently used to approximate a demonstration with a low curvature. Alternatively, a combination of a straight and a curved primitive was used to interpret an unknown curvature as in the lower left part of Fig. 5B. The average number of segmentation points (3.2) across the set suggests that when the set of primitive is insufficient to reproduce the demonstration, the algorithm compensate autonomously by increasing the number of segmentation points.

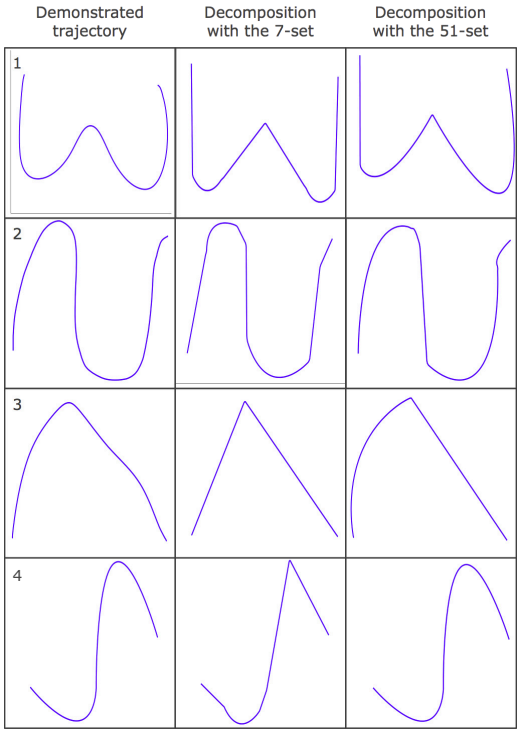

Fig. 4. Examples of decomposition with short trajectories. (First row) A hand-written W is decomposed with the sets of 7 and 51 primitives respectively. (Second row) Decomposition of a hand-written N. (Third row) Decomposition of a triangular shape recorded from an arm movement. (Fourth row) Decompositions of a model-generated trajectory formed by two consecutive primitives.

The decomposition that uses the 51-set reproduced the demonstrations accurately at a visual inspection. However, in 15% of cases, alternative solutions were found to exploit a level of redundancy. The complete test with figures is provided at the author's associate web site as support material.

Interestingly, although the decomposition with the 51-set in Fig. 5C is matching exactly the demonstrated trajectory, the reproduction with the limited 7-primitive set still results in an approximation that is more abstract and captures main features of the trajectory.

C. Decomposition of a human-generated long trajectory

The decomposition of a long human generated trajectory is shown here with both primitive sets. Fig. 6A shows the demonstrated trajectory that was recorded from human writing. To illustrate the iterative process, Fig. 6B and C show iterations two and four during decomposition with the 51-primitive set. It is possible to note that the algorithm begins by reproducing the most relevant features of

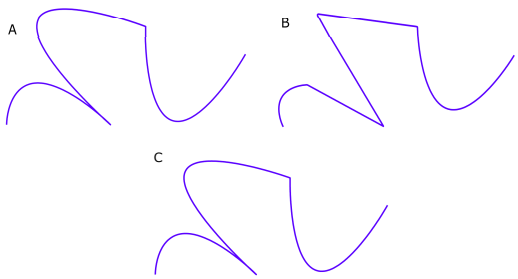

Fig. 5. Testing the decomposition on machine generated trajectories. (A) One hundred machine-generated trajectories as this one in the example are used for the extensive performance test. (B) Reproduced trajectory with the small 7-primitive sets. (C) Reproduced trajectory with the large 51-primitive set.

the demonstrated trajectory. A video showing the complete decomposition process is provided as support material. Fig. 6D shows the final approximation with the large 51-primitive set. Fig. 6E shows the reproduced trajectory as it was decomposed by the 7-primitive set. In this case, the approximation appears less accurate and straight lines are frequently used. However, the main features of the demonstration are captured indicating that the smaller set of primitives resulted in a more abstract representation. It is interesting to note that the decomposition with the larger primitive set results in better approximation with fewer parts. Although this fact appears intuitive, these experiments show that the current method achieves this tread-off that emerges autonomously when the set of primitives changes. Therefore, it can be concluded that the proposed method adapts autonomously to exploit the specific primitives, i.e. the available skills, of the agent or robotic platform that perform the movements.

The decomposition algorithm, thanks to the minimal analysis performed on the demonstration and the use of pre-learnt primitives, is particularly robust to noise. A decomposition was run on the data-set in Fig. 6A, corrupted by the addition of $\pm 1\%$ noise to each sampling point. The decomposition proceeds on this noisy data-set similarly to the case without noise. Fig. 7 shows that the reproduced trajectory is an interpretation of the noisy data. The interpretation is not noise reduction, but rather a feature extraction and reconstruction process. This simulation proves that the proposed algorithm employs its generalisation capabilities to filter noise and detect relevant features in the demonstrated trajectory.

IV. Discussion

The original idea in the proposed algorithm is to decompose an arbitrarily complex trajectory using the agent's pre-learnt primitives in a learning-by-doing, or kinesthetic, process. The process starts with a rough approximation of the demonstrated trajectory and learns step by step the features of the input data by a progressive decomposition. The final result is a sequence of primitives that is in effect an intelligent reading of a demonstrated trajectory, represented as a general and abstract concept. The strength of the algorithm

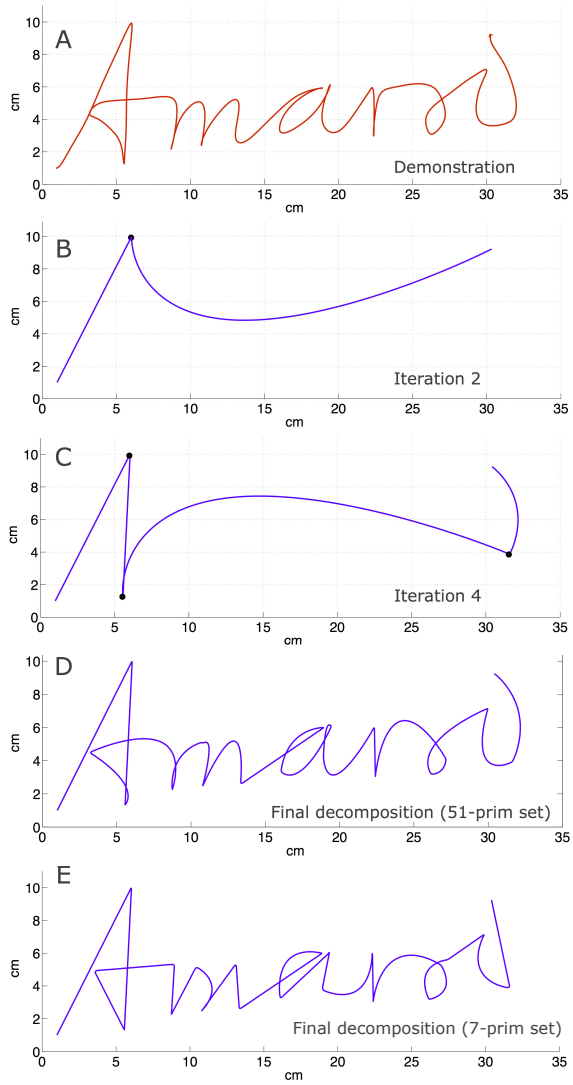

Fig. 6. Decomposition of the hand-written trajectory "Amarsi". (A) The demonstrated trajectory recorded from hand-writing. (B) Iteration 2 during the composition with the large 51-primitive set in Fig. 2B. (B) Iteration 4 during the decomposition. (C) The final decomposition and approximation using the set of 51 primitives. The demonstration was decomposed in 18 parts (D). The final decomposition and approximation using the set of 7 primitives in Fig. 2A. The algorithm decomposed the trajectory in 21 parts. Although the accuracy with the 7-primitive set is lower than with the larger set, the reproduction appears readable and to some extent a more abstract representation of the demonstration.

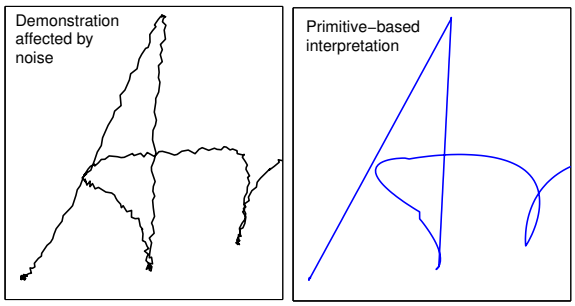

Fig. 7. Detail of the decomposition of a noisy version of the hand-written trajectory "Amarsi". (Left) Noisy demonstration. (Right) Reconstruction using the 51-primitive set.

lies in the primitive-centred and progressive search, which uses existing motor skills and implicitly solves data-induced problems like noise and discontinuous derivatives.

The algorithm appears to have generalisation capabilities even if it decomposes trajectories from one single demonstration. The generalisation capability, noticeable particularly in Fig. 4 (rows 1 and 3), derives from the interpretation of the demonstration according to the agent's set of primitives. The reconstruction from noisy data (Fig. 7) in particular shows the generalisation capability in drawing straight lines as well as maintaining cuspids. The new decomposition method is tested here using one single example for each trajectory. A promising extension is to use multiple demonstrations of the same trajectory to increase the generalisation properties of the algorithm. This study analyses a 2D scenario, however, the method can be equally applied to a 3D scenario because primitives and errors can be equally generated in 3D space, making this extension a promising venue for future research.

The proposed method focuses on the decomposition of trajectories and does not consider the learning of new primitives, which is the task of the specific movement generation algorithm chosen to work in combination with the current method. Alternative approaches could integrate the current method to learn additional primitives with experience. In fact, sequences of primitives could lead to the learning of a new longer primitive that includes two or more simpler primitives.

The variety of tasks in which simple movements are combined to achieve complex movements extends to numerous scenarios. The proposed method can be applied to those scenarios to decompose arbitrarily complex movements. The method can also be extended to recognise previously learnt sequences. This research direction includes the classification of movements for complex tasks as those in human writing. Finally, the agent-centred progressive decomposition by trial-and-error implies that the agent's skills are exploited even when decomposing trajectories with precision or complexity beyond the agent's capabilities.

V. Conclusion

A trajectory decomposition algorithm is proposed in this work. A novel idea is to start decomposing a complex trajectory with one initial single primitive and progressively increase the accuracy of the approximation through a kinesthetic process of learning-by-doing. The agent-centred process offers a new way of interpreting data as function of the agent's skills. Therefore, diverse robotic platforms with different degrees of accuracy and motor patterns can apply the method to learning progressively and autonomously the decomposition of complex trajectories. The method proves robust and displays remarkable generalisation and feature extraction capabilities, thereby proposing an intelligent tool for classifying, learning and reproducing complex movements from imprecise data. The algorithm can be applied to a variety of tasks such as imitation learning, learning of complex motor patterns, gestures, object manipulation, software-based and robotic hand-writing.

Acknowledgment

The authors would like to thank Robert Haschke for valuable comments on an early version of the manuscript. This work was supported by the European Community's Seventh Framework Programm FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics under grant agreement No 248311 - AMARSi.

Algorithm 1 Main loop

Define: Pri → A primitive from the set
Define: ME → Maximum Error (Eq. 2)
Define: InitialPoint → Initial point of the demonstration
Define: FinalPoint → Final point of the demonstration
Define: A, B → Start and end points of the sub-trajectory analysed
Define: PME → Point of maximum error
Define: DPs → Array of decomposition points
Define: IPs → Ordered list of identified primitives
Define: boxFlag → detect if trajectory leaves the areas of focus

```
A ← InitialPoint, B ← FinalPoint
[Pri, ME, PME,boxFlag] ← bestPri(A,B)
repeat
    [PriLeft, ErrLeft] ← bestPri(A,PME)
    [PriRight, ErrRight] ← bestPri(PME,B)
    if (ErrLeft < ME or ErrRight < ME or boxFlag) then
        DPs, IPs ← updateSolution(PME, DPs, PriLeft, PriRight)

        improvedSolution ← YES
        DPs, IPs ← verifyNearViaPoint(PME)
    end if
    [A, B, ME, PME] ← findSegmentToImprove
until improvedSolution == YES
```

Appendix

A. Generating the primitives with the minimum-jerk model

The minimum-jerk model [3] is used in the current study to generate both primitives for the sets and sample trajectories for the testing in Section III-B. This model plans a trajectory starting from a given starting point to a given end-point through a via-point. The constraint for planning the trajectory is to be as smooth as possible (minimum jerk). In the generated data-sets the via-point is located at the maximum of each shape, which is reached at $t = 0.5$ of the movement duration.

A demonstrated trajectory is fit by rotating and scaling the primitives accordingly to the initial (I) and final (F) points, such that the primitive fits the start and end point of the demonstration. The distance of the two trajectories (demonstration (x) and reproduction ($\hat{x}$)) is measured by interpolation of the trajectories such that they have the same movement duration with the help of cubic splines. The point-wise mean square error (MSE) used for selecting the best fitting primitive is given by

$$MSE = \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_j^i - \hat{x}_j^i)^2 \tag{1}$$

where M is the dimension of the data and N the number of samples. The point of maximum error is given by

$$ME = max(\frac{1}{M} \sum_{j=1}^{M} (x_j^i - \hat{x}_j^i)^2) \quad . \tag{2}$$

B. Algorithm in pseudo-code

Algorithm 2 verifyViaPoint(P, DPs, IPs)

```
Verify utility of neighbouring viapoints
repeat
    If a primitive is found that skips the point K (adjacent to P) and reduces the error, the point K is removed.
until a point K is removed
return DPs, IPs
```

Algorithm 3 findSegmentToImprove

```
Compute ME for all primitives in IPs from InitialPoint to FinalPoint
Find segment (A,B) with maximum error (ME) from demonstration
if intersectionExist then
    return A, B, ME, MPE
else
    END
end if
```

Algorithm 1 illustrates the main loop that starts from one single primitive and proceeds to further decompositions. Algorithm 2 is employed to exploit a newly inserted decomposition point in the attempt to eliminate neighbouring points, thereby ensuring a decompositions with fewer primitives. Algorithm 3 identifies at each iteration the part of the trajectory that requires improvement because it has the greatest discrepancy (error) with the demonstration.

REFERENCES

[1] T. Schack, "The cognitive architecture of complex movement," *International Journal of Sport and Exercise Psychology*, vol. 2, no. 4, pp. 403–438, 2004.

[2] J. Vauclair, "Phylogenetic Approach to Object Manipulation in Human and Aple Infants," *Human Development*, vol. 27, pp. 312–328, 1984.

[3] T. Flash and N. Hogan, "The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE International Conference on Robotics and Automation. Proceedings (ICRA).* , 2002.

[5] S. Schaal, "Dynamic movement primitives—a framework for motor con- trol in humans and humanoid robots," in *Proc. Int. Symp. Adaptive Motion Animals*, 2003.

[6] S.-M. Khansari-Zadeh and A. Billard, "BM: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models," in *Proceeding of the International Conference on Robotics and Automation (ICRA)*, pp. 2381–2388, 2010.

[7] R. F. Reinhart and J. J. Steil, "Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot iCub," in *IEEE-RAS International Conference on Humanoid Robots*, 2009.

[8] F. Reinhart and J.-J. Steil, "Neural learning and dynamical selection of redundant solutions for inverse kinematic control," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2011.

[9] S. Schaal and C. G. Atkeson, "Learning control in robotics – trajectory-based opitimal control techniques," *Robotics and automation magazine*, no. 2, 2010.

[10] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[11] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE international conference on robotics and automation (icra)*, 2011.

[12] R. Mann, A. D. Jepson, and T. El-Maraghi, "Trajectory segmentation using dynamic programming," in *Proceedings. 16th International Conference on Pattern Recognition.* , vol. 1, pp. 331 – 334, 2002.

[13] S. Hellbach, J. P. Eggert, E. Koerner, and M.-H. Gross, "Basis decomposition of motion trajectories using spatio-temporal nmf," in *International Conference on Artificial Neural Networks, Proceedings of the*, 2009.

[14] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen, "Constructing skill trees for reinforcement learning agents from demonstration trajectories," in *Advances in Neural Information Processing Systems 23 (NIPS).*, 2010.

[15] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[16] J. Kohlmorgen and S. Lemm, "A Dynamic HMM for On-line Segmentation of Sequential Data," in *NIPS: Advances in Neural Informaiton Processing Systems*, 2001.

[17] V. Mohan, P. Morasso, J. Zenzeri, G. Metta, V. S. Chakravarthy, and G. Sandini, "Teaching a humanoid robot to draw 'Shapes'," *Autonomous Robots*, vol. 31, no. 1, pp. 21–53, 2011.

[18] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila, "Primitives for smoothing mobile robot trajectories," *Robotics and Automation, IEEE Transactions on*, 1995.

[19] T. Kulvicious, K. Ning, M. Tamosiunaite, and F. Woergoetter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *Robotics, IEEE Transactions on*, vol. 28, pp. 145–157, 2012.

[20] R. Sosnik, B. Hauptmann, A. Karni, and T. Flash, "When practice leads to co-articulation: the evolution of geometrically defined movement primitives," *Experimental Brain Research*, vol. 156, pp. 422–438, 2004.

[21] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, 2011.

[22] C. Atkeson and J. McIntyre, "Robot trajectory learning through practice," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, 1986.